ECE 474A/57A
Computer-Aided Logic Design

Branch-and-Bound and Simulated Annealing

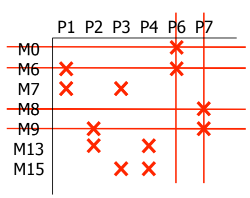
Logic Optimization Techniques

- Logic Optimization Techniques
  - K-maps (Graphical)
  - Quine-McCluskey (Exact Algorithm)
    - Tabular Minimization
    - Row/Column Dominance
  - Espresso (Heuristic) – we'll see this one soon
- Other Generalized Algorithms
  - **Branch-and-bound**
  - Simulated Annealing
  - many more exists …
    - Integer Linear Programming (ILP)
    - Dynamic Programming
    - Genetic Algorithms

- Very general algorithm – can be applied to a variety of problems
- Based on the idea of a decision tree
  - Varies in that it tries to visit only part of the tree

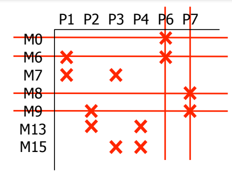
Decision Trees

- Decision tree
  - Enumeration approach in which we have *n* decision variables, and list the 2^n possible values

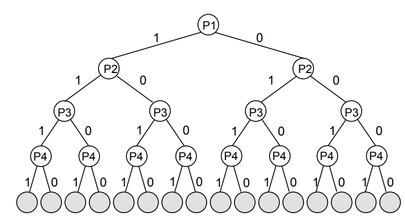


Given a prime implicant chart and the corresponding essential prime implicants, how do we derive a minimum cover with the remaining prime implicants?

Decision Trees



Remove the essential prime implicants, they are already in the cover.
Number the remaining prime implicants so it's easier for us to read.



Let's start our decision tree. What are the decision to make?

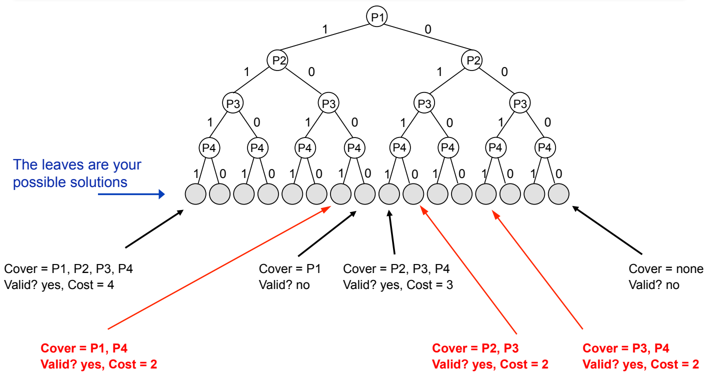
Should we include P1 in our cover?

Should we include P2 in our cover?

Should we include P3 in our cover?

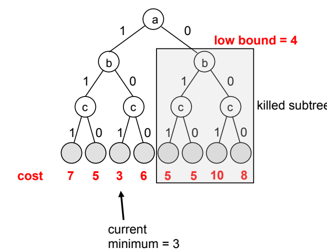
Should we include P4 in our cover?

Decision Trees



Branch-and-bound Idea

- Branch-and-bound
  - Several optimal solutions may exist, we only need to find one
  - Idea is that maybe we only have to visit part of the decision tree
  - If we can estimate the low bound to a subtree, and that low bound is higher than the current minimum, we don't need to look at that subtree



Generic Branch-and-bound Pseudocode

```
BCP( F, U, currentSol ){
    ( F, currentSol ) = REDUCE( F, currentSol )

    if( terminalCase( F ) ){
        if( cost( currentSoln ) < U ){
            U = cost( currentSoln )
            return ( currentSoln )
        }
    }

    L = LOWER_BOUND( F, currentSoln )
    if ( L ≥ U ) return ( "no solution" )

    xi = CHOOSE_VAR( F )
    S1 = BCP( Fxi, U, currentSol ∪ {xi} )

    if( cost( S1 ) = L ) return ( S1 )
    S0 = BCP( Fxi', U, currentSoln )

    return BEST_SOLUTION( S1, S0 )
}
```

- REDUCE: Iteratively finds essential variables and applies row/column dominance to simplify matrix – updates current solution with these changes
- terminalCase: Is currentSoln a valid solution? If valid and better than existing solution, update solution and cost
- LOWER_BOUND: Calculate the lower bound of the subtree to see if the subtree is worth looking at
- CHOOSE_VAR: Make a decision – which prime do we want to include/exclude?
  - no effect on correctness, help with efficiency of runtime
- S^1 = BCP(...): Recursive call on subtree that includes the prime
- If S^1 subtree contained the low cost solution ignore S^0 subtree, otherwise recursive call on subtree excluding the prime
- Return best solution

Branch-and-bound Pseudocode

- Initial call to BCP
  - currentSoln set to empty
  - Upper bound (U) set to the number of decisions (prime implicants) + 1
    - Guarantees that the first valid solution found will be accepted
  - F is the current constraint equation
- Call to REDUCE(F)
  - Try to simplify the matrix by recursively
    - Removing essential columns and adding it to currentSoln
    - Remove dominating rows
    - Remove dominated columns
    - Continue until matrix is empty, or problem is cyclic
- Splitting Variable x_i
  - Variable selection has no impact on correctness, impacts run time
    - Find a good solution fast so upper bound is close to optimal solution and more pruning can occur
  - Potential candidates?
    - Column that covers many rows is more likely to be part of optimal solution
    - Column that covers many short rows since short rows have a lower chance of being covered

Branch-and-bound – Lower Bound Calculation

- How do I calculate the lower bound of a subtree?
  - Varies depending on your problem
  - Minimum cover problem
    - lower bound = number of prime implicants (columns committed so far) + MIS
- Maximally Independent Set (MIS)
  - Equal to the number of independent rows in the table
    - Rows are independent if no overlapping X's
    - Indicates the lowest possible number of prime implicants required to cover the remaining minterms
  - We want worst case, so we pick the largest set
  - If no independent rows are found, the lower bound for a cyclic matrix is at least 2
    - If matrix cyclic no column covers all rows (which would have enabled reduction of matrix)
    - Thus, a minimum of two columns are required to cover all rows

	p1	p2	p3	p4
1	X	X		
2			X	X
3	X		X	
4		X		X
5	X			X
6		X	X	

{1, 2}

{3, 4}

{5, 6}

MIS = 2

Finding MIS

MIS_QUICK Heuristic

- Simple algorithm can be used to find MIS
  - || M || denotes rows left in M after deleting rows intersecting with row i
  - CHOOSE_SHORTEST_ROW subprocedure can be done in several ways
    - Option 1 - Row i is row with the fewest nonzero columns, breaking ties in lexicographical order
    - Option 2 - Row i is selected by column counts of its columns, breaking ties in lexicographical order
      - Does a better job finding larger MIS

```
MIS_QUICK(M){
   MIS = Φ
   do {
      i = CHOOSE_SHORTEST_ROW(M)
      MIS = MIS ∪ {i}
      M = DELETE_INTERSECTING_ROWS(M, i)
   } while ( || M || > 0 )
   return MIS
}
```

	P1	P2	P3	P4
m1	X	X	X	
m2		X	X	
m3	X		X	X

Option 1

$w_1 = 3$
$w_2 = 2$
$w_3 = 3$

w_1 calculated by adding all x's in row 1

Option 2

$w_1 = 7$
$w_2 = 5$
$w_3 = 6$

P1 has x's in column 1, 2, and 3

w_1 calculated by adding all x's in column 1, 2, 3

MIS_QUICK Example

- Use MIS_QUICK (option 1) to find MIS

MIS = Φ

	P1	P2	P3	P4	P5	P6	
m1	X				X		$w_1 = 2$
m2	X	X		X			$w_2 = 3$
m3		X	X				$w_3 = 2$
m4				X	X	X	$w_4 = 3$
m5			X	X			$w_5 = 2$
m6		X				X	$w_6 = 2$
m7	X		X				$W_7 = 2$

i = 1 MIS = {1} Add row 1 to MIS
Delete intersecting rows (2, 7, 4)

	P1	P2	P3	P4	P5	P6	
m3		X	X				$w_3 = 2$
m5			X	X			$w_5 = 2$
m6		X				X	$w_6 = 2$

i = 3 MIS = {1,3} Add row 3 to MIS
Delete intersecting rows (5,6)

MIS = {1, 3}
Low bound = 0 + 2 (no essentials previously added)

MIS_QUICK Example

- Use MIS_QUICK (option 2) to find MIS

MIS = Φ

	P1	P2	P3	P4	P5	P6	
m1	X				X		$w_1 = 5$
m2	X	X		X			$w_2 = 9$
m3		X	X				$w_3 = 6$
m4				X	X	X	$w_4 = 7$
m5			X	X			$w_5 = 6$
m6		X				X	$w_6 = 5$
m7	X		X				$W_7 = 6$

i = 1 MIS = {1} Add row 1 to MIS
Delete intersecting rows (2, 7, 4)

	P1	P2	P3	P4	P5	P6	
m3		X	X				$w_3 = 4$
m5			X	X			$w_5 = 3$
m6		X				X	$w_6 = 3$

i = 5 MIS = {1, 5} Add row 5 to MIS
Delete intersecting rows (3)

	P1	P2	P3	P4	P5	P6	
m6		X				X	$w_6 = 2$

i = 6 MIS = {1, 5, 6} Add row 6 to MIS
Matrix empty – Done!

MIS = {1, 5, 6}
Low bound = 0 + 3 (no essentials previously added)

Option 2 found a larger MIS set which leads to higher lower bound (i.e. more pruning)

Branch-and-bound

Example 1

- Using Branch-and-bound find minimum cover

F = Φ
U = 6+1 = 7

L = 2

	P1	P2	P3	P4	P5	P6
m1	×		×		×	
m2	×			×		×
m3		×	×			×
m4		×		×	×	

No reduction can be made, matrix cyclic

Call to BCP(F, U, {})

1. Initialize best solution (F) and current cost (U) variables
2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK returns {m1}, but matrix is cyclic so MIS is at least 2
   Lower bound (L) = # prime implicants + MIS = 0 + 2 = 2
5. L ≥ U? No.
6. x_i = P1
7. S^1 = BCP(F_{xi}, U, currentSoln ∪ x_i)

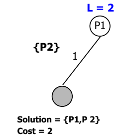
Branch-and-bound

Example 1

Call to BCP(F_{P1}, U, {P1})

~~F = Φ~~
~~U = 6+1 = 7~~
F = {P1, P2}
U = 2

L = 2



	P1	P2	P3	P4	P5	P6
~~m1~~	×		×		×	
~~m2~~	×			×		×
m3		×	×			×
m4		×		×	×	

P1 included - covers m1, m2

	P2	P3	P4	P5	P6
m3	×	×			×
m4	×		×	×	

No row dominance
P2 dominates P3, P4, P5, P6

	P2
m3	×
m4	×

P2 becomes essential - add to F

Matrix empty

2. Reduce matrix
3. Solution found? Yes.
   cost(currentSoln) < U ?
   cost({P1, P2}) < 7 ? Yes.

   Update placeholders

Branch-and-bound

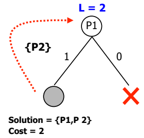
Example 1

Call to BCP(F, U, {})

F = {P1, P2}
U = 2

	P1	P2	P3	P4	P5	P6
m1	×		×		×	
m2	×			×		×
m3		×	×			×
m4		×		×	×	

L = 2



1. Initialize best solution (F) and current cost (U) variables
2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK returns {m1}, but matrix is cyclic so MIS is at least 2
   Lower bound (L) = # prime implicants + MIS = 0 + 2 = 2
5. L ≥ U? No.
6. x_i = P1
7. S^1 = BCP(F_{xi}, U, currentSoln ∪ x_i) ← **Returns from here with updated F and U**
8. Cost(S^1 = L)? Yes.
   Kill S^0 subtree.

Done! All options examined.

Branch-and-bound

Example 2

- Using Branch-and-bound find minimum cover

F = Φ
U = 11+1 = 12

L = 4

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
m1	1	1									
m2		1	1								
m3			1	1							
m4	1			1							
m5					1	1				1	
m6						1	1		1		
m7							1	1			
m8						1		1		1	1
m9					1				1	1	1
m10					1			1	1		
m11					1		1				1
m12	1										1
m13					1	1		1			

No reduction can be made, matrix cyclic

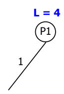
BCP(F, U, {})

1. Initialize best solution (F) and current cost (U) variables
2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK returns {m1, m3, m5, m7}
   lower bound (L) = # prime implicants + MIS = 0 + 4 = 4
5. L ≥ U? No.
6. x_i = P1
7. S^1 = BCP(F_{xi}, U, currentSoln ∪ x_i)

Branch-and-bound
Example 2

BCP(F_{P1}, U, {P1})

F = Φ
U = 11+1 = 12



2. Reduce matrix

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
~~m1~~	1	1									
m2		1	1								
m3			1	1							
~~m4~~	1			1							
m5					1	1				1	
m6						1	1		1		
m7							1	1			
m8						1		1		1	1
m9					1				1	1	1
m10					1			1	1		
m11					1		1				1
~~m12~~	1										1
m13					1	1		1			

P1 included - covers m1, m4, m12

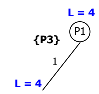
	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
m2	1	1								
m3		1	1							
m5				1	1				1	
m6					1	1		1		
m7						1	1			
m8					1		1		1	1
m9				1				1	1	1
m10				1			1	1		
m11				1		1				1
m13				1	1		1			

No row dominance
P3 dominates P2, P4

Branch-and-bound
Example 2

BCP(F_{P1}, U, {P1})

F = Φ
U = 11+1 = 12



2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK returns {m7, m5}
   lower bound (L) = # prime implicants + MIS = 2 + 2 = 4
5. L ≥ U? No.
6. x_i = P5
7. S^1 = BCP(F_{xi}, U, currentSoln ∪ x_i)

	P3	P5	P6	P7	P8	P9	P10	P11
~~m2~~	1							
~~m3~~	1							
m5		1	1				1	
m6			1	1		1		
m7				1	1			
m8			1		1		1	1
m9		1				1	1	1
m10		1			1	1		
m11		1		1				1
m13		1	1		1			

P3 becomes essential – only one to cover m2, m3. Add to currentSoln

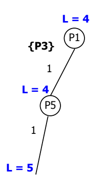
	P5	P6	P7	P8	P9	P10	P11
m5	1	1				1	
m6		1	1		1		
m7			1	1			
m8		1		1		1	1
m9	1				1	1	1
m10	1			1	1		
m11	1		1				1
m13	1	1		1			

No reduction can be made, matrix cyclic

Branch-and-bound
Example 2

BCP(F_{P1P5}, U, {P1, P5})

F = Φ
U = 11+1 = 12



2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK returns 2 (No independent sets)
   lower bound (L) = 3 + 2 = 5
5. L ≥ U? No.
6. x_i = P6
7. S^1 = BCP(F_{xi}, U, currentSoln ∪ x_i)

	P5	P6	P7	P8	P9	P10	P11
~~m5~~	1	1				1	
m6		1	1		1		
m7			1	1			
m8		1		1		1	1
~~m9~~	1				1	1	1
~~m10~~	1			1	1		
~~m11~~	1		1				1
~~m13~~	1	1		1			

P5 included – covers m5, m9, m10, m11, m13

	P6	P7	P8	P9	P10	P11
m6	1	1		1		
m7		1	1			
m8	1		1		1	1

No row dominance.
P7 dominates P9, P8 dominates P10, P11

	P6	P7	P8
m6	1	1	
m7		1	1
m8	1		1

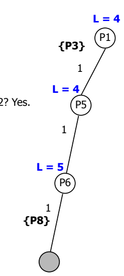
No reduction can be made, matrix cyclic

Branch-and-bound
Example 2

BCP(F_{P1P56}, U, {P1, P5, P6})

~~**F = Φ**~~
~~**U = 11+1 = 12**~~

F = {P1, P3, P5, P6, P8}
U = 5



Solution = {P1, P 3, P5, P6, P8}
Cost = 5

2. Reduce matrix
3. Solution found? Yes.
   cost({P1, P3, P5, P6, P8})<12? Yes.
   Update placeholders

	P6	P7	P8
~~m6~~	1	1	
m7		1	1
~~m8~~	1		1

P6 included – covers m6, m8

	P7	P8
m7	1	1

P8 dominates P7

	P8
m7	1

P8 becomes essential – add to F

matrix empty

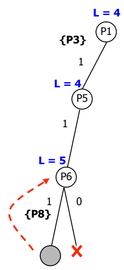
Branch-and-bound
Example 2

BCP(F_{P1P5}, U, {P1, P5})

F = {P1, P3, P5, P6, P8}
U = 5

	P6	P7	P8
m6	1	1	
m7		1	1
m8	1		1

2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK returns 2 (No independent sets)
   lower bound (L) = 3 + 2 = 5
5. L ≥ U? No.
6. x_i = P6
7. S^1 = BCP(F_{x_i}, U, currentSoln ∪ x_i)
8. Cost(S^1 = L)? Yes.
   Kill S^0 subtree.



Solution = {P1, P 3, P5, P6, P8}
Cost = 5

Returns from here with updated F and U

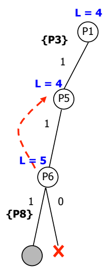
Branch-and-bound
Example 2

BCP(F_{P1}, U, {P1})

F = {P1, P3, P5, P6, P8}
U = 5

	P5	P6	P7	P8	P9	P10	P11
m5	1	1				1	
m6		1	1		1		
m7			1	1			
m8		1		1		1	1
m9	1				1	1	1
m10	1			1	1		
m11	1		1				1
m13	1	1		1			

2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK returns 2 (No independent sets)
   lower bound (L) = 3 + 2 = 5
5. L ≥ U? No.
6. x_i = P5
7. S^1 = BCP(F_{x_i}, U, currentSoln ∪ x_i)
8. Cost(S^1 = L)? No.
9. S^0 = BCP($F_{x_i'}$, U, currentSoln ∪ x_i')



Solution = {P1, P 3, P5, P6, P8}
Cost = 5

Returns from here with updated F and U

Branch-and-bound
Example 2

BCP($F_{P1P5'}$, U, {P1, P5'})

F = {P1, P3, P5, P6, P8}
U = 5

	P5	P6	P7	P8	P9	P10	P11
m5	1	1				1	
m6		1	1		1		
m7			1	1			
m8		1		1		1	1
m9	1				1	1	1
m10	1			1	1		
m11	1		1				1
m13	1	1		1			

P5 excluded

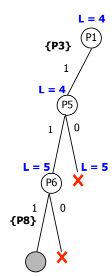
	P6	P7	P8	P9	P10	P11
m5	1				1	
m6	1	1		1		
m7		1	1			
m8	1		1		1	1
m9				1	1	1
m10			1	1		
m11		1				1
m13	1		1			

m8 dominates m5, m13

	P6	P7	P8	P9	P10	P11
m5	1				1	
m6	1	1		1		
m7		1	1			
m9				1	1	1
m10			1	1		
m11		1				1
m13	1		1			

No reduction can be made, matrix cyclic

2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK = {m5, m10, m11}
   lower bound (L) = 2 + 3 = 5
5. L ≥ U? Yes.
   Kill S^0 subtree.



Solution = {P1, P 3, P5, P6, P8}
Cost = 5

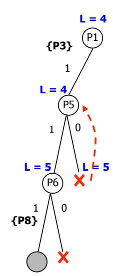
Branch-and-bound
Example 2

BCP(F_{P1}, U, currentSoln ∪ {P1})

F = {P1, P3, P5, P6, P8}
U = 5

	P5	P6	P7	P8	P9	P10	P11
m5	1	1				1	
m6		1	1		1		
m7			1	1			
m8		1		1		1	1
m9	1				1	1	1
m10	1			1	1		
m11	1		1				1
m13	1	1		1			

2. Reduce matrix
3. Solution found? No.
4. Calculate lower bound on subtree
   MIS_QUICK returns 2 (No independent sets)
   lower bound (L) = 3 + 2 = 5
5. L ≥ U? No.
6. x_i = P5
7. S^1 = BCP(F_{x_i}, U, currentSoln ∪ x_i)
8. Cost(S^1 = L)? No.
9. S^0 = BCP($F_{x_i'}$, U, currentSoln ∪ x_i')
10. return BEST_SOLUTION(S^1, S^0)



Solution = {P1, P 3, P5, P6, P8}
Cost = 5

Returns from S^0 here, S1 subtree has better solution

Branch-and-bound

Example 2

BCP(F, U, {})

2. Reduce matrix

3. Solution found? No.

4. Calculate lower bound on subtree

   MIS_QUICK returns {m7, m5}

   lower bound (L) = # prime implicants + MIS = 2 + 2 = 4

5. L ≥ U? No.

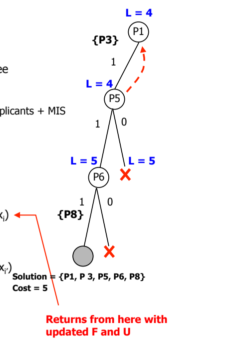
6. x_i = P1

7. S^1 = BCP(F_{xi}, U, currentSoln ∪ x_i)

8. Cost(S^1 = L)? No.

9. S^0 = BCP($F_{xi'}$, U, currentSoln ∪ x_i')

F = {P1, P3, P5, P6, P8}
U = 5



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
m1	1	1									
m2		1	1								
m3			1	1							
m4	1			1							
m5					1	1				1	
m6						1	1		1		
m7							1	1			
m8						1		1		1	1
m9					1				1	1	1
m10					1			1	1		
m11					1		1				1
m12	1										1
m13					1	1		1			

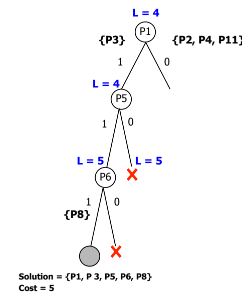
Branch-and-bound

Example 2

Call to BCP($F_{P1'}$, U, {P1'})

2. Reduce matrix

F = {P1, P3, P5, P6, P8}
U = 5



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
m1	1	1									
m2		1	1								
m3			1	1							
m4	1			1							
m5					1	1				1	
m6						1	1		1		
m7							1	1			
m8						1		1		1	1
m9					1				1	1	1
m10					1			1	1		
m11					1		1				1
m12	1										1
m13					1	1		1			

P1 excluded

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
~~m1~~	1									
~~m2~~	1	1								
~~m3~~		1	1							
~~m4~~			1							
m5				1	1				1	
m6					1	1		1		
m7						1	1			
~~m8~~					1		1		1	1
~~m9~~				1				1	1	1
m10				1			1	1		
~~m11~~				1		1				1
~~m12~~										1
m13				1	1		1			

P2, P4, P11 becomes essential – only one to cover m1, m4, m12 (respectively)

Branch-and-bound

Example 2

Call to BCP($F_{P1'}$, U, {P1'})

2. Reduce matrix

3. Solution found? No.

4. Calculate lower bound on subtree

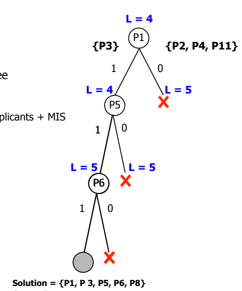
   MIS_QUICK returns {m5, m7}

   lower bound (L) = # prime implicants + MIS = 3 + 2 = 5

5. L ≥ U? Yes.

   Kill subtree

F = {P1, P3, P5, P6, P8}
U = 5



Done! All options examined.

	P5	P6	P7	P8	P9	P10
m5	1	1				1
m6		1	1		1	
m7			1	1		
m10	1			1	1	
m13	1	1		1		

No row dominance
P5 dominates P10

	P5	P6	P7	P8	P9
m5	1	1			
m6		1	1		1
m7			1	1	
m10	1			1	1
m13	1	1		1	

m13 dominates m5

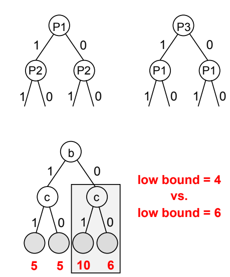
	P5	P6	P7	P8	P9
m5	1	1			
m6		1	1		1
m7			1	1	
m10	1			1	1

No reduction can be made, matrix cyclic

Branch-and-Bound Summary

- Branch-and-Bound algorithm used to help determine a minimal cover
  - We have a set of possible prime implicants to choose from (i.e. P1, P2, P3, P4)
- Which one should we choose first?
  - Methods to choose splitting variable – we skipped
  - Solution still optimal, maybe just slower
- Determining the lower bound is very important
  - We want to be accurate so we don't waste our time
  - However, this step should still be fast
- Additionally, as prime implicants are added, we can use row/column dominance to try and simplify remaining matrix
  - Helps to speed up algorithm
- Solution is exact (optimal), running time varies on selection process and bounding calculation

			0	6	7	8	9	13	15
	~~(2,6)~~	~~0-10~~	×	×					
	~~(8,9)~~	~~-001~~				×	×		
P1	(6,7)	011-		×	×				
P2	(9, 13)	10-1					×	×	
P3	(7, 15)	-111			×				×
P4	(13, 15)	1-11						×	×

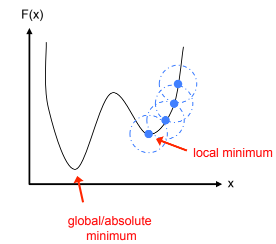


Logic Optimization Techniques

- Logic Optimization Techniques
  - K-maps (Graphical)
  - Quine-McCluskey (Exact Algorithm)
  - Espresso (Heuristic)
- Other Generalized Algorithms
  - Branch-and-bound
  - **Simulated Annealing**
  - many more exists …
    - Integer Linear Programming (ILP)
    - Dynamic Programming
    - Genetic Algorithms

Simulated Annealing - Background

- Simulated Annealing
  - Name and inspiration come from annealing in metallurgy
  - Heating and controlled cooling of a material to reduce defects/increase strength
- Applied to local search methodology to avoid getting stuck at the local minimum



General Simulated Annealing Pseudocode

```
Simulated_Annealing{
    S = initial solution
    T = initial temperature (>0)
    while( T > 0 ){
        S' = pick a random neighbor to S
        C = cost of S – cost of S'
        if( C > 0 ){
            S = S'
        }
        else{
            r = random number in range [0…1]
            m = 1/e|C/T|
            if( r < m ){
                S = S'
            }
        }
        T = reduced T;
    }
}
```

Derive a new solution S', by randomly making a change to the current solution

Determine the cost difference between the old and new solution. (Is the new solution better?)

If the new solution is better, keep the new solution

Randomly, we sometimes take the worse solution (AVOID LOCAL MINIMUM)

The probability of this happening corresponds the to the temperature, the higher the temperature (early in the algorithm) the more likely we take this chance

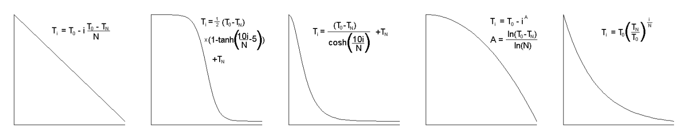
e = mathematical constant 2.71828…

Decrease the temperature

This is the cooling schedule – how fast does the temperature decrease?

Simulated Annealing – Cooling Schedules

- Choosing initial temperature and cooling schedule has great impact on the algorithm
  - Make sure we run long enough to find a good solution
  - Make sure we get out of local optimum (take chances on worse solutions)
- Many options available, no definitive way to choose these



Simulated Annealing Cooling Schedules
Brian T. Luke, Ph.D.
http://fconyx.ncifcrf.gov/~lukeb/simanf1.html

Simulate Annealing – Example

- How do we apply to the minimum cover problem?

1. Choose an initial solution, set an initial temperature
2. Is temperature T > 0? Yes
3. Make a random change to S

   What can we change?
   - Adding another prime implicant to our cover
   - Removing a prime implicant from the current cover ✓ **Remove P2**
4. Determine cost difference

   C = cost of S – cost of S'
   C = 4 – 3 = **1**

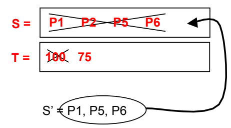
   We should also consider if this solution is correct. (Yes)

   a) Is the solution better? Yes.

   Keep new solution
6. Decrease Temperature

   T = T - 25 = 75

	m1	m2	m3	m4
P1	✖	✖		
P2			✖	✖
P3	✖		✖	
P4		✖		✖
P5	✖			✖
P6		✖	✖	



Simulate Annealing – Example

2. Is temperature T > 0? Yes
3. Make a random change to S

   What can we change?
   - Adding another prime implicant to our cover ✓ **Add P4**
   - Removing a prime implicant from the current cover
4. Determine cost difference

   C = cost of S – cost of S'
   C = 3 – 4 = **-1**

   We should also consider if this solution is correct. (Yes)

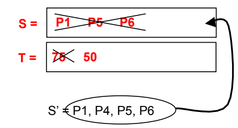
   a) Is the solution better? No.

   b) Should we randomly accept it anyways?

   r = 0.215 (random number), m = $1/e^{|-1/75|}$ = 0.9867
   0.215 < 0.9867? Yes. Keep new solution
7. Decrease Temperature

   T = T - 25 = 50

	m1	m2	m3	m4
P1	✖	✖		
P2			✖	✖
P3	✖		✖	
P4		✖		✖
P5	✖			✖
P6		✖	✖	



Simulate Annealing – Example

1. Is temperature T > 0? Yes
2. Make a random change to S

   What can we change?
   - Adding another prime implicant to our cover
   - Removing a prime implicant from the current cover ✓ **Remove P1**
3. Determine cost difference

   C = cost of S – cost of S'
   C = 4 – 3 = **1**

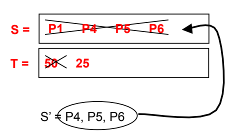
   We should also consider if this solution is correct. (Yes)

   a) Is the solution better? Yes.

   Keep new solution
5. Decrease Temperature

   T = T - 25 = 25

	m1	m2	m3	m4
P1	✖	✖		
P2			✖	✖
P3	✖		✖	
P4		✖		✖
P5	✖			✖
P6		✖	✖	



Simulate Annealing – Example

1. Is temperature T > 0? Yes
2. Make a random change to S

   What can we change?
   - Adding another prime implicant to our cover ✓ **Add P1**
   - Removing a prime implicant from the current cover
3. Determine cost difference

   C = cost of S – cost of S'
   C = 3 – 4 = **-1**

   We should also consider if this solution is correct. (Yes)

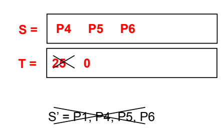
   a) Is the solution better? No.

   b) Should we randomly accept it anyways?

   r = 0.978 (random number), m = $1/e^{|-1/25|}$ = 0.961
   0.978 < 0.961? No. Disregard new solution
6. Decrease Temperature

   T = T - 25 = 0

	m1	m2	m3	m4
P1	✖	✖		
P2			✖	✖
P3	✖		✖	
P4		✖		✖
P5	✖			✖
P6		✖	✖	



Simulate Annealing – Example

1. Is temperature T > 0? No

Done!
Solution : P4, P5, P6

	m1	m2	m3	m4
P1	✖	✖		
P2			✖	✖
P3	✖		✖	
P4		✖		✖
P5	✖			✖
P6		✖	✖	

S = P4 P5 P6

T = 0

- Is this solution optimal?
  - No
- Ideally, this algorithm would run longer so we can explore more of the solution space and possible find a better solution

Conclusion

- Considered several logic optimization techniques
  - K-maps
  - Quine-McCluskey
  - Espresso
- Considered several other generalized algorithms useful for logic optimization as well as other applications
  - Branch-and-bound
  - Simulated Annealing
  - Many more exists …
    - Integer Linear Programming (ILP)
    - Dynamic Programming
    - Genetic Algorithms